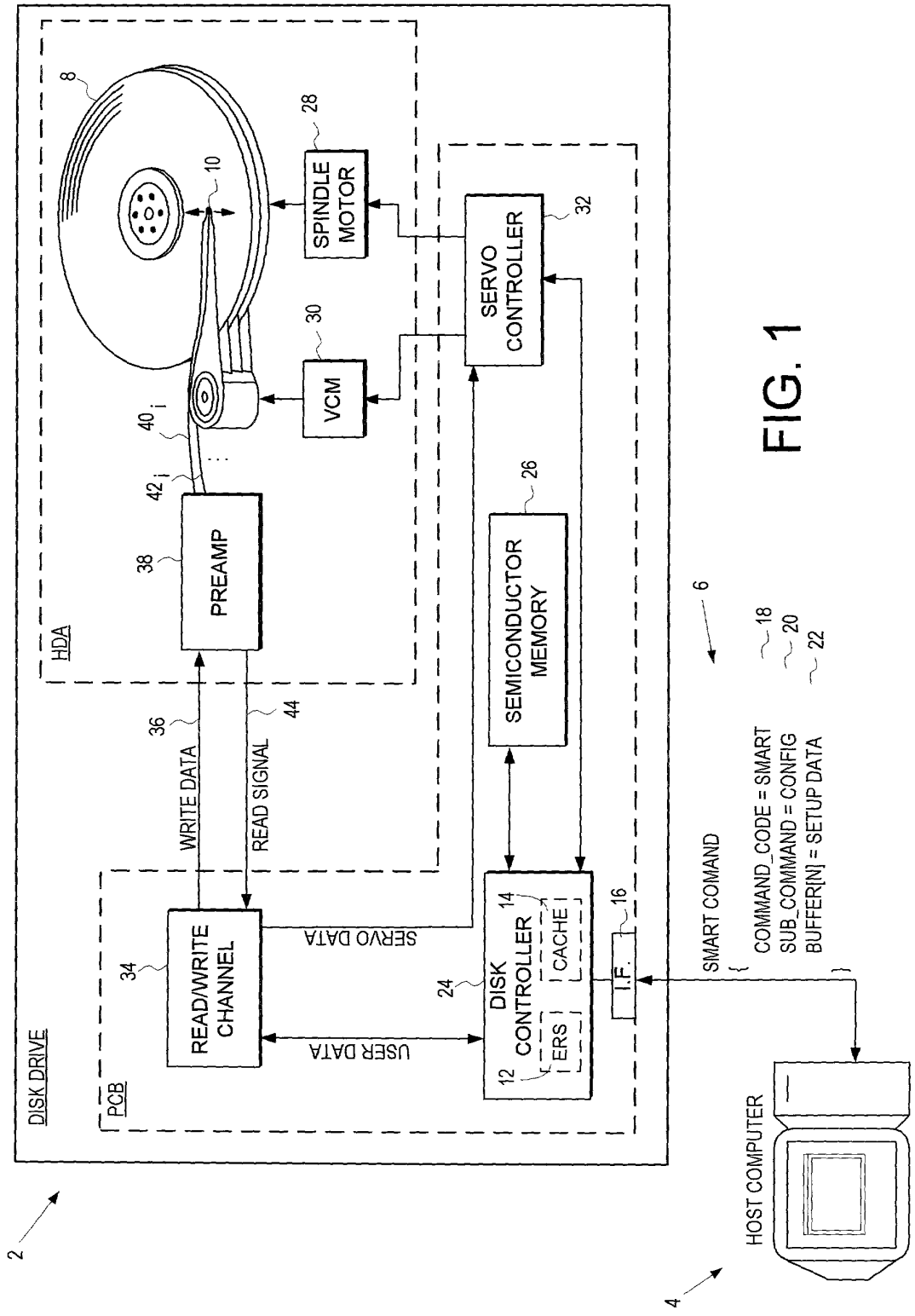
DISK DRIVE
PCB
HDA
8
10
28
SPINDLE MOTOR
30
VCM
40 i
42 i
38
PREAMP
32
SERVO CONTROLLER
26
SEMICONDUCTOR MEMORY
36
WRITE DATA
44
READ SIGNAL
34
READ/WRITE CHANNEL
SERVO DATA
USER DATA
24
DISK CONTROLLER
12
ERS
CACHE
14
16
I/F
6
SMART COMAND
{
COMMAND_CODE = SMART
SUB_COMMAND = CONFIG
BUFFER[N] = SETUP DATA
}
18
20
22
HOST COMPUTER
2
4

FIG. 1

```
BOOL DeviceIoControl(hDevice, dwIoControlCode, lpvInBuffer, cbInBuffer,
                     lpvOutBuffer, lpcBytesReturned, lpoOverlapped)

HANDLE          hDevice;          // hande of device
DWORD           dwIoControlCode;  // control code of operation to perform
LPVOID          lpvInbuffer;      // address of buffer for input data
DWORD           cbInBuffer;       // size ofinput buffer
LPVOID          lpvOutBuffer;     // address of output buffer
DWORD           cbOutBuffer;      // size of output buffer
LPDWORD         lpcBytesReturned; // address of actual bytes of output
LPOVERLAPPED    lpoOverlapped;    // address of overlapped structure
```

FIG. 2A

```
typedef struct _SENDCMDINPARAMS {
   DWORD     dwBufferSize   // Size of bBuffer in bytes
   IDEREGS   irDriveRegs;   // Structure with drive register values.
   BYTE      chDriveNumber; // Physical drive number to send command to (0,1,2,3).
   BYTE      chReserved[3]; // Reserved for future expansion.
   DWORD     dwReserved[4]; // Reserved for future expansion.
   BYTE      chBuffer[1];   // Buffer of arbitrary length in which to store the data to be written to drive.
} SENDCMDINPARAMS, *PSENDCMDINPARAMS, *LPSENDCMDINPARAMS;
```

FIG. 2B

```
typedef struct SendCmdOutParams {
   DWORD          dwBufferSize;  // Size of bBuffer in bytes
   DRIVERSTATUS   DriverStatus;  // Driver status structure.
   BYTE           chBuffer[1];   // Buffer of arbitrary length in which to store the data read from
                                 // the drive.
} SENDCMDOUTPARAMS, *PSENDCMDOUTPARAMS;
```

FIG. 2C

```
typedef struct _IDEREGS {
   BYTE      chFeaturesReg;      // Used for specifying DFP sub commands.
   BYTE      chSectorCountReg    // IDE sector count register
   BYTE      chSectorNumberReg   // IDE sector number register
   BYTE      chCylLowReg         // IDE low order cylinder value
   BYTE      chCylHighReg        // IDE high order cylinder value
   BYTE      chDriveHeadReg      // IDE drive/head register
   BYTE      chCommandReg;       // Actual IDE command.  Checked for validity by driver.
   BYTE      chReserved;         // reserved for future use.  Must be zero.
} IDEREGS, *PIDEREGS;
```

FIG. 2D

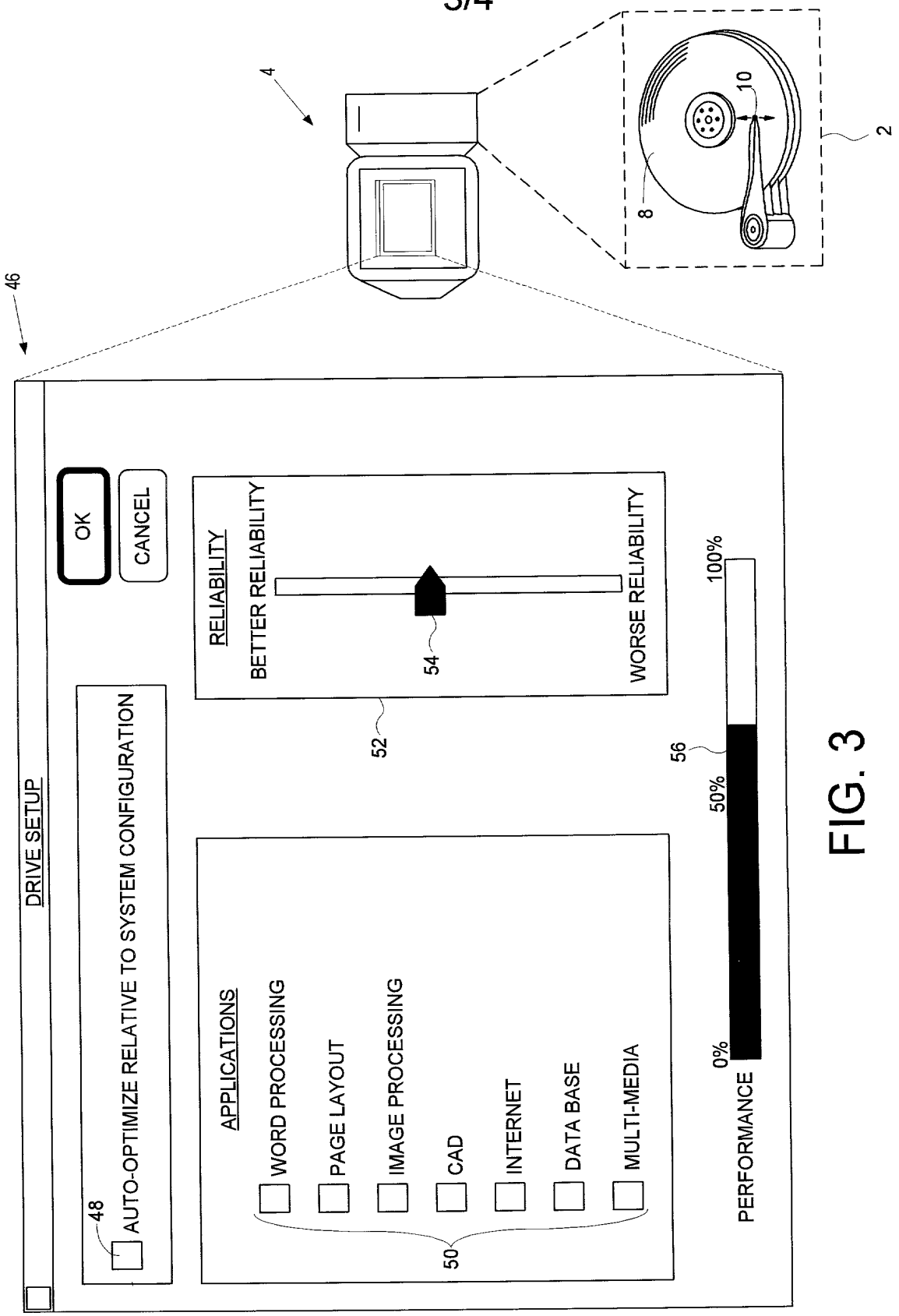

FIG. 3

TITLE: DISK DRIVE FOR RECEIVING SETUP DATA IN A SELF MONITORING ANALYSIS AND REPORTING TECHNOLOGY (SMART) COMMAND
INVENTOR: MICHAEL S. ROTHBERG
APPLICATION NUMBER: K35A0840

FIG. 4